Insider Threats: Identifying Anomalous Human Behaviour in Heterogeneous Systems Using Beneficial Intelligent Software (Ben-ware)

Andrew Stephen McGough, David Wall, John Brennan, Georgios Theodoropoulos, Ed Ruck-Keene
Durham University
United Kingdom
{stephen.mcgough, d.s.wall, j.d.brennan, georgios.theodoropoulos, e.a.ruck-keene}
@durham.ac.uk

Budi Arief, Carl Gamble, John Fitzgerald, Aad van Moorsel
Newcastle University
United Kingdom
{budi.arief, carl.gamble, john.fitzgerald, aad.vanmoorsel}
@newcastle.ac.uk

Sujeewa Alwis
Insighlytics Ltd
United Kingdom
sujeewa@insighlytics.com

ABSTRACT

In this paper, we present the concept of "Ben-ware" as a beneficial software system capable of identifying anomalous human behaviour within a 'closed' organisation's IT infrastructure. We note that this behaviour may be malicious (for example, an employee is seeking to act against the best interest of the organisation by stealing confidential information) or benign (for example, an employee is applying some workaround to complete their job). To help distinguish between users who are intentionally malicious and those who are benign, we use human behaviour modelling along with Artificial Intelligence. Ben-ware has been developed as a distributed system comprising of probes for data collection, intermediate nodes for data routing and higher nodes for data analysis. This allows for real-time analysis with low impact on the overall infrastructure, which may contain legacy and low-power resources. We present an analysis of the appropriateness of the Ben-ware system for deployment within a large closed organisation, comprising of both new and legacy hardware, to protect its essential information. This analysis is performed in terms of the memory footprint, disk footprint and processing requirements of the different parts of the system.

Categories and Subject Descriptors

H.3.4 [**Information storage and retrieval**]: Systems and Software—*User profiles and alert services*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms

Management, Security, Human Factors, Theory

Keywords

Insider threats; detection; anomalous behaviour; human behaviour; artificial intelligence; assistive tool; ethics.

1. INTRODUCTION

In a large closed organisation, where the security of the information systems is a key asset, it is desirable to identify quickly when a user (a legitimate employee) is acting in an anomalous manner, especially when this behaviour is against the prescribed practices of the organisation. This might indicate that they have become an insider threat and acting maliciously or that someone else is using their credentials [2]. The types of anomalous behaviour we seek to identify are those where a user is acting outside of the rules of conduct set out by the organisation. For example, this may be to steal confidential information, place false information within the system, or alter internal records. The identification of malicious behaviour is complicated by the fact that other users may perform actions outside of the organisation's rules without malicious intent [8, 17], e.g. over-riding security systems to expedite a task. We therefore use human behaviour modelling along with Artificial Intelligence (AI) [12, 14, 16] to identify malicious users from the benign system users.

The current state of the art on insider threat detection and prevention includes using a suite of complementary monitoring and auditing techniques [3]; combining structural anomaly detection with modelling of psychological factors for identifying potential insiders [4]; examining behavioural characteristics of potential insiders to distinguish between malicious and benign behaviours [5]; creating a decision support system through a 10-step program to maximise the efficiency of the organisation's analyst [10]; and better understanding through a multidisciplinary approach [11].

Other approaches have addressed issues such as misuse of legitimate access to documents or resources [1]. Honeypots (fake resources within an organisation) have been proposed as a countermeasure to insider threats [13], however, these only work when the user is unaware of the significance of the resource they seek. Greitzer and Hohimer [6] see that the best chance of insider detection comes from the collection of multiple data sources, both computer and human factors based [17]. However, we go further to offer a solution that uses Machine-Learning to identify anomalous activity based on a broad range of data sources. Kandias *et al.* [7] use psychometric tests as part of their prediction model for identifying insider threats. We see psychometric tests as a potential data source, in addition to other relevant human factors information, such as any Human Resources (HR) risk information. Thompson [15] uses Hidden Markov Models to identify divergence between normal and insider threat patterns, this shares similarities with our approach.

Existing IT based approaches in identifying anomalous behaviour of users within a system tend to work through a centralised data collection approach in which case the information is held and processed on a single dedicated server. If the organisation has more than one site, either all sites share one central server (necessitating the need for a large server and high-bandwidth networking) or each site has its own server (meaning that a user's behaviour on different sites is not correlated). This is especially significant if parts of the organisation becomes detached from the network and/or may roam between sites (e.g. a laptop). We overcome these problems and the need for a central server by using a *distributed infrastructure* consisting of lightweight probes hosted on each computer, intermediate nodes which can cache and forward data and high-end nodes each of which can process the records from a small subset of users.

Moving the machine learning to individual hosts would require each host to have substantial computational capacity – unlikely in an organisation with networks of heterogeneous legacy resources – and could lead to losing the ability to detect anomalies across the whole organisation. We overcome this by exploiting the higher performance computers within the organisation to process a small number of users each, thus reducing the centralised impact.

Another key feature is the use of human behavioural analysis allowing for common patterns of user behaviour and their individual behavioural signature to be identified, and anomalies against these detected; this approach reduces the number of false positives. This work has been made possible through an interdisciplinary collaboration among computing science, criminology, and behavioural analysis experts. Although systems exist which use AI in order to identify user anomalies, to our knowledge we are the first to use human factors and analysis of human behaviour to influence and develop AI techniques.

Conventional AI approaches to classification problems are not well suited to the detection of insider threats. This is because classification using machine learning works by training on a data set in which each piece of data is tagged as either being 'good' or 'bad' – working best when there are equal proportions of 'good' and 'bad' data. However, in the case of insider threats – where thankfully the incidences of malicious activity against the organisation are low – it is not possible to obtain a balanced training set. Instead we employ here an AI approach of categorising 'normal' behaviour and look for outliers from this behaviour.

Finally it should be noted that no computer-based solution can detect criminal activity within a system. It can, at best, detect breeches in prescribed practices – identifying potential threats; and reducing the false-positives through the use of AI and human-factors. In organisations, there is a need to understand when cyber security issues (threats and risks) turn into cybercrime problems (actual harm) for the organisation. One of the problems with identifying potential risks and harms is that of throwing up false positives – Ben-ware helps reduce false positives by highlighting those who are more likely to be a criminal threat, enabling security personnel to focus resources where they matter.

The rest of this paper is organised as follows. The next section explores the background to the Ben-ware approach, which is then followed by the vision for Ben-ware. The paper then lays out implementation and performance characteristics of the Ben-ware prototype we have developed, followed by a discussion of the results and context of the work, before presenting conclusions and future directions.

2. BACKGROUND

One of the primary concerns of organisations is in determining when insiders are threatening effective working and causing financial, informational or reputational loss. The main challenge is that there are different types of insider threat, users are not always malicious and the good guy / bad guy binary classification does not neatly apply. It should be noted that we focus on 'closed' organisations where employees are contractually bound to adhere to strict rules of employment and information security.

Employees within an organisation can be classified within a two-dimensional space – of actions and intent. Actions range from good to bad; likewise intentions range from good to bad. Employees can be placed within this space. Loch *et al.* [8] stated that intent is a binary state of accidental or intentional; however, we argue that this is a continuum (from 0% good (bad) to 100% good), likewise for actions. Criminals with financial or espionage motivations can be seen as having bad actions and bad intentions, whist in the opposite corner (good actions and good intentions) are well-behaved individuals who abide by the rules. Unfortunately employees can occupy any point within this space. Based on this classification and the work by Wall [17], we have identified six main categories of users:

1. **The well-behaved:** Abides by the rules set out within the organisation and works without malicious intent. Both intention and actions are >90% good.
2. **The negligent:** Does not realise they are breaking the rules. Rules are broken in order to 'make life easier' either for themselves or others within the organisation. Although their actions are bad their intentions are good. These staff are often the most susceptible to social engineering attacks. Actions are <50% good, intentions are >50% good.
3. **The ambitious:** Is aware of the fact that they are breaking the rules but do so as this gives them an advantage. Similar to the negligent user their actions are bad (<50%) but their intentions are good (>50%). The main distinction between these groups is the ambitious user will actively and knowingly break the rules.

4. **The malicious insider:** Is actively trying to thwart the organisation (regardless of motivation). This user has bad actions and bad intentions (<20%).
5. **The whistle-blower:** Although this category can be seen as part of malicious insider, we separate it out here to highlight the point that this user has bad intentions (<20%) from the organisation's point of view, however, they have good intentions (>90%) from their own point of view. Actions are however, bad (<25%).
6. **The sleeper:** This user has bad intentions (<20%) though at present does not exhibit bad actions (>80%).

2.1 Scenario

In order to illustrate the complexity of detecting malicious insiders – especially in dealing with the matter of false positives – let us consider the following four hypothetical cases of users that illustrate the range of diverse insider threats:

- **Theft of data (whistleblowing/ revenge)** – Joanne Ben-Nevis has been denied promotion by her upmarket clothing manufacturer employer, she wishes to break with the organisation to go freelance, and simultaneously embarrass her employer by exposing some of the organisation's dirty linen to the public – a large collection of files relating to the third-world locations and conditions at outsourcing companies used for manufacturing – practices that are legal, but would in the current political climate seriously damage the commercial reputation of the organisation. During the final three months of her employment she slowly, deliberately and regularly downloads the files that she wants. *Characteristics: Large data acquisition but over a (reasonably) long period of time.*
- **Espionage (Trade Secret theft)** – Paul Penyghent is also moving on and wishes to steal the contact details of clients. He also wishes to take the design of a vacuum cleaner that the company is manufacturing. He knows that once he has told his employer of his intention to leave he will be placed on gardening leave with no further access to the system. *Characteristics: Large data acquisition over a very short period of time.*
- **Corruption of data (falsifying accounts)** – Ben Lawers wants to change his data in his HR file. Many years ago he made an inappropriate approach to a female colleague, which is on his record. He believes this is why he has been passed up for promotion. He sits next to his line manager and keeps hoping that she will leave her terminal unlocked. *Characteristics: The use of someone else's account to perform a malicious act.*
- **Circumvention of procedure (Well-meaning)** – Jane Galtymore (CEO) is frustrated by the security procedures of the company that she runs and routinely takes files out of her company's security system so that she can easily work on the train commute and also at home. *Characteristics: Although breeching security, the acts are not malicious.*

These cases illustrate the key problems in insider threats, some are malicious and others not, but each is still a threat to the organisation. It is also particularly hard to identify intent with enough strength to distinguish maliciousness. Therefore, in order to reduce the inevitable false positives and the associated human resources fallout, it is important to introduce additional elements into the mix (e.g. human factors information) that may help the software understand the user in order to rate the threat or risk of their anomalous behaviour.

Although the above scenarios depict both malicious and non-malicious activity, it may be desirable for the organisation to 'deal' with all of these users in some way. The negligent user may need further training in order to understand the implications of their actions, whilst the ambitious user may need senior staff to point out the long-term consequences of not adhering to policy.

Even though the AI can 'learn' each individual users profile over a period of time, this does not help in the short-term. A malicious employee who has stolen a single file every day since starting employment will be seen as performing this as their 'normal' work-pattern. Likewise, false-positives could be generated when an employee changes roles – if Ben Lawers becomes a line-manager he may suddenly start to access personnel records.

With Ben-ware we seek to bridge this gap by ingesting 'human factors' information into the AI to help it understand when the threats and risks will become harmful. Such information is often already held in organisations and would consist of hard facts about individuals based upon a combination of:

- HR information
- Verifiable intelligence sources, say, relevant personal knowledge held by a line manager
- Personal characteristic information, e.g. psychometric indicators (Belbin or Myers-Briggs test)
- Determination of a user's IT skills [9] to understand individual's profiles
- General profile of the individual's occupation.

3. OUR VISION: BEN-WARE

Our vision is to devise a novel architecture for detecting anomalous user behaviour with minimal impact to the existing system – which may comprise of legacy and low-performance computers – in terms of requiring no extra facilities, transparency to the users, and the ability to cope with unreliable, low bandwidth network links and parts of the network which may become isolated for periods of time.

The architecture (Figure 1) consists of a hierarchy of services where the base entities (those without subordinates) are hosts to be monitored, while the higher entities form the control structure that collates the gathered information. Each host runs Ben-ware probes which gather information on user activity. A Mid-Level Controller (MLC) is an autonomous intelligent agent, running on a host possessing

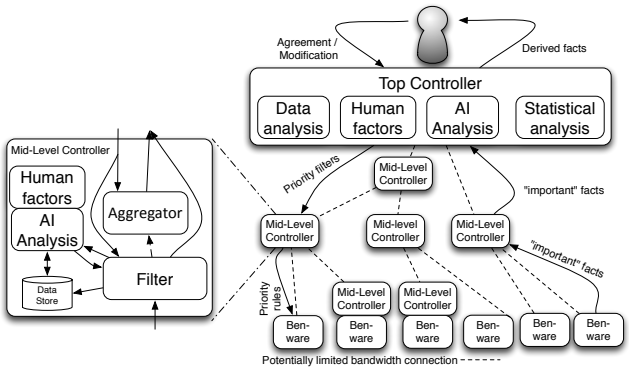

Figure 1: Overall Architecture

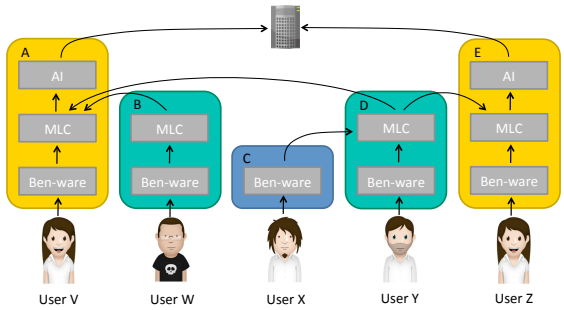

Figure 2: Deployment and communication chains

enough processing power, controlling a group of lower level entities. The MLC may have an AI engine within it, in which case the AI will be responsible for analysing one (or more) users in order to identify anomalies – distributing workload over the existing system. If the MLC does not have an AI, it will just act as a filter and aggregator of data, which will be sent to another MLC/AI responsible for the particular user – e.g. computer B in Figure 2 has only an MLC which forwards to the MLC/AI on computer A. The Top Controller (TC) is an MLC that can present collated information about user's activities to a human manager, who can either take action or indicate that this is not malicious activity. Although a single TC is shown, the system could have multiple TCs allowing human managers at different levels/locations to monitor users for whom they are responsible.

Ben-ware probes can capture information such as user logins/logouts, USB attachments, web access and file transfers. This information is transferred up to an MLC for further processing. The MLC filters messages based on a set of filter rules, allowing it to store the messages for later aggregation or send them immediately to the AI responsible for that user. The MLC also keeps a historical log of all messages for the last n days (n is configurable), allowing for forensic analysis of the raw data at a future stage. Messages sent to the TC will trigger an alert to a human manager about a potential security threat. Filtering and aggregation of messages sent within the system will reduce communication overheads and improve the ability to prioritise information that should be immediately propagated up for further inspection/action, thus allowing for a more adaptable infrastructure coping better with low and intermittent networking, as well as partitioned networks.

We identify three deployment scenarios (Figure 2):

- **Machine with Probes only (C):** This may be a low-power/legacy computer incapable of running the full service
- **Machine with Probes and MLC, without AI (B and D):** A more powerful computer that is capable of running the MLC. In this case the computer may be powerful enough to run an AI though there may not currently be a need for running one
- **Machine with Probes, MLC and AI (A and E):** This is a powerful computer running all three services.

There is also the scenario of an external computer, not owned by the organisation, coming in through a VPN connection or a Bring Your Own Device (BYOD). In this case, there will be no Ben-ware on the device and only limited monitoring can be provided at the interface between the organisation and the device (for example, connecting to the network, opening a shared drive or copying a file over the network). This scenario is not considered further here.

The AI responsible for a particular user can modify the filtering and aggregation rules for that user on any MLC. The rules determine which types of information should be propagated up and how much aggregation of the information should be performed. For example, the AI may determine that a particular user rarely logs in before 8am and thus indicates that a user login before 8am should be immediately passed up. However, any logins during their 'normal' hours should only be aggregated and sent up once per day. Thus, information can be prioritised allowing for better use of low-bandwidth networks.

The AI for a particular user will be located on an MLC/AI 'close' to the normal login location for that user. The location of the MLC/AI will be chosen by ordering computers based on their network connectivity speed to the user's normal computer and taking the first one with enough computational power to run the AI[1], helping reduce network traffic. In the case of network partitioning, if the MLC/AI for a user and the MLC for the computer where the user has logged in are in different partitions then messages will initially be stored for sending after the network becomes re-connected. However, if the partitioning lasts for a significant amount of time, determined to be greater than the time used for aggregating messages (normally one day), then a new MLC/AI will be allocated to that user and the data will be processed there. This will lead to having two MLC/AIs responsible for a given user when the network comes together again. In which case one of the MLC/AIs will become the master and the other will send all data to the master before stopping.

Although – for the purposes of management and reporting – the MLCs are connected in a tree structure, the MLCs are also able to communicate in a more peer-to-peer manner as outlined in Figure 2. In this case, when a user logs into a machine running Probes/MLC (say computer B), the MLC for that Probe will locate the MLC/AI (say computer A) responsible for that user (using a DNS-like lookup service) and will obtain the filter and aggregation rules for that user. For computers running Probes but no MLC (say computer C), there will already be an MLC assigned to it (say computer D), which will carry out the same actions on behalf of the Probes. Note that this information may be cached on an MLC if the user regularly logs into that machine.

Data will be held in a database system within an MLC connected to the Probe where the data was produced, unless it is being cached during transit. This data will be held on the MLC for a set amount of time, after which it will be destroyed. Holding the data allows for forensic investigations in the aftermath of an attack, or in order to help determine why the system flagged up a particular user.

Ben-ware uses its AI component (described further in Section 4.4) to detect anomalies in users' activities, which in turn might indicate insider attacks. However, it is possible that a user could circumvent this by always behaving in a 'bad' manner from the start, causing the AI to see this as a normal behaviour of that user. Nonetheless, such behaviour might be detected by other means, such as monitoring by a line manager (i.e. human intervention).

[1]The selection of the computer on which to run the AI is a much more complex problem, compounded by locations of other AIs and a user having multiple 'normal' computers.

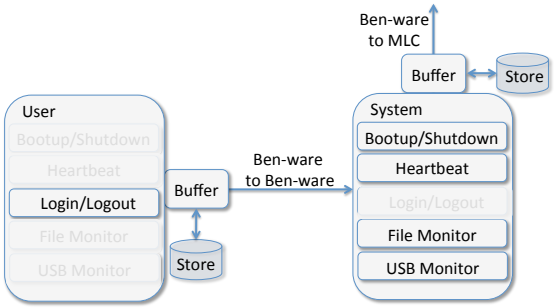

Figure 3: User and System Ben-ware probes

4. IMPLEMENTATION

We have implemented a proof of concept Ben-ware solution as a set of Java programs running on a Windows platform (capable of running on XP, Win7, Win8). In this section, we detail the four main components (Ben-ware probes, Mid-Level Controllers, the Artificial Intelligence, and Top Controllers/User Interface).

4.1 Event types

We first define the event types as these are passed between the different agents within the system, albeit with modifications based on the level of communication. A non-exhaustive list of event types is listed in Table 1. Each of these events can have supplementary data. For example, *Rename* will contain a copy of the original file name and the new name whilst *HTTP* will contain the URL.

Table 1: Event types

ID	Name	Description
1.1	Boot up	The host boots up
1.2	Shut down	The host is shut down
2.1	Heartbeat	The system is still active
3.1	Logon	User logs into system
3.2	Logout	User logs out of system
4.1	Copy	File copy (internal, USB)
4.2	Create	File creation (internal, USB)
4.3	Move	File move (internal USB)
4.4	Delete	File deletion (internal, USB)
4.5	Modify	File modify (internal, USB)
4.6	Open	File open (internal, USB)
4.7	Close	File close (internal, USB)
4.8	Rename	File rename (internal, USB)
5.1	HTTP	HTTP request performed
6.1	USBInsert	USB Insertion
6.2	USBRem	Removal of USB device

4.2 Ben-ware Probe

The Ben-ware probes are executed as two instances of the same program: one deals with actions related to the User logged in to the computer, while the other handles general System actions. These are shown in Figure 3. Upon booting up, Windows creates the System instance of the Ben-ware probe. This instance contains the probes that report when the host boots up/shuts down, and provides the heart beat messages (used to detect if someone is trying to attack Ben-ware) to the MLC. It is also responsible for monitoring access to designated areas of the local file system and the insertion/removal of USB memory sticks. Due to the Windows security model, the Ben-ware System instance is not able to detect when a user logs in/out or obtain the username. To address this, a second instance of the Ben-ware probes is started whenever a user logs in. This User instance informs the System instance of the identity of the logged in user and sends periodic heart beat messages indicating the user remains logged in.

To reduce the network overhead, the System instance is responsible for transmitting messages to the MLC and also backing-up any unsent messages to the local disc. User instance messages for the MLC are forwarded through the System instance.

4.3 Mid-Level Controller (MLC)

The MLC consists of three components: server, database and aggregator, as shown in Figure 4. The server component is responsible for listening on a TCP port for incoming data from the System probes. The received data is checked for consistency and, if appropriate, added to the database. The database used here is the open-source MySQL server allowing long-term storage of probe data and complex queries to be performed. The aggregator component queries the database for entries within a specified time frame, and computes a set of aggregate records indicating the type of action performed and the number of occurrences (this is done by compiling a list of all event types associated to the user and tallying up the number of times each event type is invoked by this user in the specified time frame). The aggregates are sent to the AI at pre-defined intervals (e.g. once per day), unless the user is deemed to have exceeded a pre-defined threshold (determined by the AI) in which case that total is sent immediately. If a connection to the AI is unavailable then the aggregate data is stored for later sending.

The MLC takes its input from three sources: a configuration file read on start-up, data received from the user level probes, and user related rules sent from the AI. The AI rule-sets define limits for the maximum number of a particular event type which can happen in a time period before the MLC should immediately flag this up to the AI (for example, the number of files copied to a USB stick), hence allowing faster response of the system to potential threats. The configuration file contains all the parameters needed to communicate with other components and connect to a database.

Messages from the probes are delivered to the MLC via a TCP connection, as comma-separated strings, consisting of a code defining the type of event (Table 1), the source computer, the user who was logged in, the time the event took place, the priority of the event, the operating system and an extension element for specific data for that event type (e.g. the name of a file when copying to USB). These messages are checked for correct formatting, then stored in the database for later analysis.

The MLC also receives user rule change messages from the AI. Each message is a triplet of rule type, those users affected by the rule and a threshold for the rule. For example, a

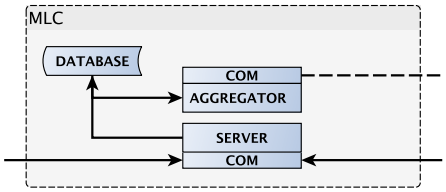

Figure 4: MLC architecture

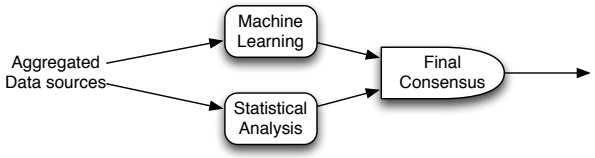

Figure 5: Overview of the intelligent agent

triple of <*USB Inserts greater than, Joanne, 3*> indicates that if Joanne inserts more than 3 USB sticks in one day, this should be immediately flagged up.

All data is then analysed to determine the total number of events for each user/host combination. As a minimum, the MLC will generate daily aggregate reports of individual user behaviour sent to the AI. These aggregates will actually be calculated on an n-minute basis (default 60) and compared against the user rules supplied by the AI. If a user has exceeded a user rule's limit then the aggregate information will be sent to the AI immediately.

4.4 Artificial Intelligence

The AI component implements an anomaly detection algorithms to detect suspicious activities by users, which is achieved using a combination of machine learning and statistical algorithms to create the intelligent agent. The two algorithms first independently assess the data arriving from the MLC before forming a consensus (as shown in Figure 5), which is mapped to a risk score (i.e. NO RISK, MODERATE RISK, HIGH RISK or VERY HIGH RISK).

The inputs into the AI include login patterns, file use patterns, web access patterns, use of external storage devices, human factors, along with additional information regarding the liveliness of a device derived from a 'heart beat'. This information is then used to generate the features required for the anomaly detection algorithms. This feature set includes the following:

- **Logon:** Numbers of: logons, hosts logged on, out of hours logons, logons on user's host and logons on shared/other hosts
- **External Storage Devices:** Numbers of: device accesses, out of hour device accesses, hosts used for device accesses, device accesses on own hosts and device accesses on other hosts
- **File Use:** Numbers of: file accesses, file writes, file writes on to external devices, sensitive file writes, sensitive file writes on to external devices, out of hours file accesses, out of hours file writes, out of hours file writes on to external devices, out of hours sensitive file writes, out of hours sensitive file writes on to external devices, hosts used for file accesses, file accesses on own hosts and file accesses on other hosts
- **HTTP:** Numbers of: http requests, out of hours http requests, http requests on black-listed/categorised sites.

Through the use of these metrics we are able to develop a profile of a user's normal activities, allowing us to detect anomalous actions which differ from these activities. It should be noted that this set of features is not exhaustive nor are the base probes. Both can be extended in order to monitor other types of user activity.

All features were normalised using the 95th percentile in order to eliminate distortions caused by features of different degrees of variability. The 95th percentile was chosen since it helped avoid normalisation by outliers. The other candidates considered were the maximum value, median, mean and the 90th percentile.

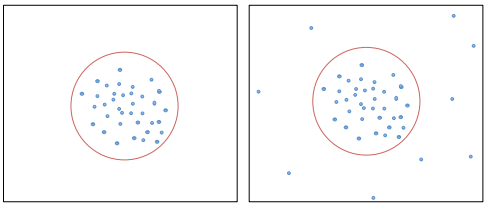

Figure 6: Bounding circle for a set of points

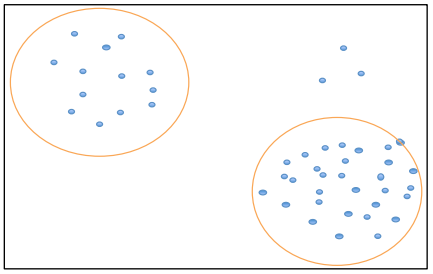

Figure 7: Clusters of user behaviour

We see here that a number of approaches to machine-learning could be deployed depending on the types of anomalous behaviour we are trying to detect and the feature sets which we have available. Thankfully the number of incidences of insider threats within most (if not all) organisations is very small. Likewise the lack of examples of suspicious behaviours within any potential dataset, the fact that not all abnormal behaviour represents malicious intent, and the lack of categorisations make the use of conventional categorisation approaches unsuitable.

A one-class classifier called Support Vector Data Description (SVDD) [12, 14] was chosen as the machine-learning algorithm. It adopts a similar approach to the widely used Support Vector Machines (SVM) [16] in calculating the minimum bounding hyper-sphere for the training dataset. Figure 6 shows two examples of these boundaries – (a) is the case where a compact bounding can be achieved, whilst for (b) there exists outliers. The only parameter that needs fine-tuning by the operator during training is the maximum fraction of inputs to be excluded from the bounding surface. Although a hyper sphere is the simplest form of the boundary that can be obtained using this method, complex forms of boundaries can also be constructed using kernel methods. For example, a radial basis function can be used as the kernel. This requires adjusting another parameter, the width of the kernel. However, over-optimisation of this configuration using a relatively small dataset could lead to over-fitting, which can be avoided by using cross-validation methods.

By using SVDD, potential malicious users can be identified through their anomalous activities, which will come up as outliers. In comparison, well-behaved users tend to have their activities clustered closely within the boundaries.

One of the key observations during this work was that user behaviour might not always be consistent over a period of time. This may be due to role change or performing multiple roles. It may be more effective to construct multiple boundaries to enclose different 'profiles' (Figure 7). A clustering method (e.g. K-means clustering) can be used to obtain clusters of different behaviour and then use SVDD to train a separate classifier for each cluster. A significant improvement was obtained using this modification and we

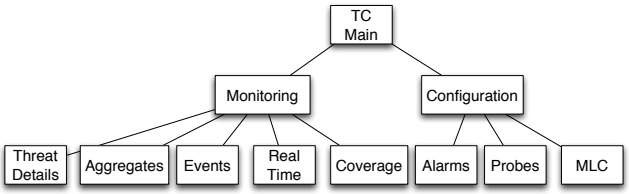

Figure 8: Top Controller organisation

believe this approach is novel for anomaly detection, but the detail is beyond the scope of this paper.

This algorithm can be used with different combinations of features to identify different types of insider threat scenarios. We trained the algorithm to capture scenarios related to 'stealing' files, though our approach could easily be retrained for other threats. During feature selection, a cross-validation method was used. It incrementally added features and retaining those delivering performance improvements. The retained features were the number of file writes on to external devices, number of sensitive file writes, number of sensitive file writes on to external devices, number of out of hours file writes on to external devices, number of out of hours sensitive file writes and number of out of hours sensitive file writes on to external devices.

The statistical methods utilised the same features as separate indicators. The difference was that they were utilised in a single-dimensional space in comparison to the multi-dimensional space used for the machine-learning. Each incoming data point was compared against the percentile values calculated using the training data set and a score was assigned based on the amplitude of the indicators. In order to capture weak signals (for example, someone stealing a smaller number of files over a period of time) both accumulated amplitude values and the duration of continuous behaviour were considered.

4.5 Top Controller / User Interface

The Top Controller (TC) provides a user interface to the Ben-ware system. Although this has not been fully implemented, we have spent some time developing a prototype interface specification. The TC is designed to provide a broad overview of the system, presenting the operator – who may be a computer administrator or line manager – with a view of each of the (managed) users in the system with a traffic light indicator highlighting the level of concern for that user (green for low concern through to red for high concern). Figure 8 illustrates how the different interfaces can be organised for the TC, which is broken down into two main areas of monitoring and configuration.

At any time, the operator can monitor the information and data held within the system. This might be to investigate a concern over a user or just to see what activities a particular user has been performing. The details of particular threats can be investigated along with the aggregates on which these threats are based. The operator may also choose to dig further into the data held about a particular user by looking at the specific events or coverage details (such as the proportion of files within the system the user has downloaded). In all cases this information can be presented graphically or in the form of tables. A configuration interface is also built into the TC allowing the operator to modify how the whole Ben-ware system is operating.

Here we present a small number of user-interface mock-ups in order to exemplify the proposed interface. Figure

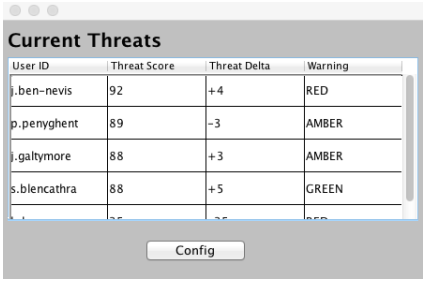

Figure 9: Initial screen showing current threats

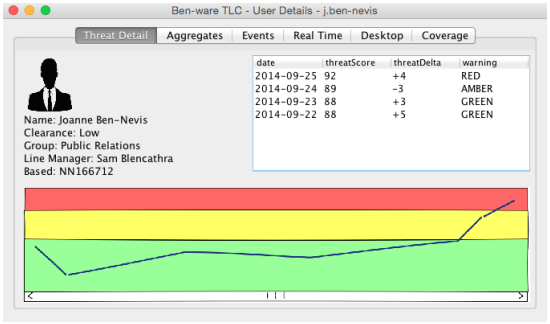

Figure 10: Details of a user over recent days

9 shows the initial screen for the TC, which allows rapid identification of potential threats. Each of the users can be clicked on in order to dive into the details for that user. Figure 10 illustrates the threat scores for a particular user while Figure 11 shows the data on individual events carried out by the user. This information – along with the aggregated data – can be used by the operator as part of initial verification of the user's activity.

4.6 Synthetic User generation

Due to complexity of collecting live data from users with the Ben-ware system – most notably the requirement to deploy the system for many months – it was decided to generate a synthetic trace log of user interaction with the system. The trace log was generated using a Markov state change model providing the probability for a user to move from any given activity to any other activity. A set of fictitious users was developed: Jo, Petra, Scott, Sally, Ben and Mel. These users were each given their own user profile.

- **Jo:** ambitious person – wants to impress and will do anything within reason to gain an advantage, starts early and leaves late, often takes work home, likes to be in meetings and has typical access rights.
- **Petra:** journey-person – a chief exec, sees little chance of promotion, puts in little effort, works 9-5, never stays late, never works from home, attends lots of meetings to waste time, and has typical access rights.
- **Scott:** Frontline secretary – low paid on sharp end, works 9-5, never stays late, never works from home, low level of security access, and hardly ever attends meetings.
- **Sally:** all round good employee – happy with job, likes to help others, often proxies for IT support, works 9-5 but with large variance, avoids meetings, won't work from home but will work late, and is often found on other people's computers, sorting out problems.
- **Ben:** tech support – performs all actions to keep IT working, works 8-6 with high variance, low chance of

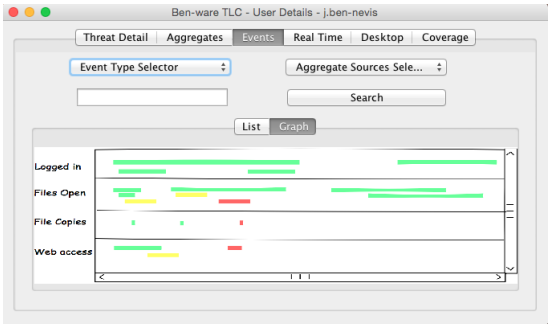

Figure 11: Individual events on a time-line

meetings, will work late, but if working from home won't take files with him and changes his working computer a lot.

- **Mel:** PA to Chief Executive – works 8:30-5:30 (arrive before boss and leave after), has to attend lots of meetings with boss, often needs to stay late or take work home to finish.

Each user's day is assumed to consist of meetings, breaks, lunchtime and working time. Working time is assumed to be on one of the tasks of reading, writing, Internet browsing, converting (documents), admin and personal admin – with each user having their own probability for moving between activities, for attending meetings and taking (lunch) breaks. It is assumed that a user – at the end of their working day – will either stay late to finish their current activity, take the activity home (requiring the copying of files to a USB stick – and a corresponding upload the following morning) or just leave the activity until the following day.

For each user, 18 months of trace data was generated. The first 12 months was 'good' activity – where no user performed data theft – whilst in the last six months users were allowed to go 'bad' in one of four different ways:

- **Bad1:** a lot of files were stolen on a single day.
- **Bad2:** a small number of files (usually 2) were stolen on each day for a period of time.
- **Bad2_gaps:** we modified the Bad2 datasets to insert gaps in continuous file stealing patterns. For example after five days of continuous file stealing, the user does not steal files for two days.
- **Bad3:** an increasing number of files were stolen on each day for a period of time.

In all cases, the date at which the user goes 'bad' is chosen at random. We acknowledge here the bias that may be present due to developing test data for our own system. We minimised this effect by having two teams: one who developed the data and the second who developed the AI.

5. RESULTS

We evaluate the Ben-ware solution both in terms of its detection accuracy and also its resource usage on the legacy hardware used for development and testing. The latter relates to the performance impact of having these components installed on the host, to demonstrate that the Ben-ware solution does not degrade the overall performance.

5.1 Probes

As the Ben-ware probes can be deployed on legacy (low power) hardware, we evaluate the probes in terms of impact on CPU load, Memory, Disk space and Network.

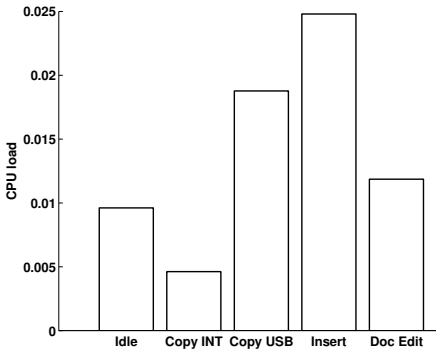

Figure 12: Average CPU increase with Ben-ware

5.1.1 *CPU impact*

To find out how the probes might affect performance, we experimented with running the probes on several different machines. In all cases there was no appreciable performance impact to the user. We conducted further tests on the lowest specification laptop used, a 12 year old Sony VAIO PCG-FR, with an AMD mobile Athlon XP 2000+ running at 1.67GHz, 256MB RAM, to determine CPU impact. As all other systems had higher specifications, we see these results as a worst-case scenario. The machine was running Windows XP Home, Service Pack 3, that was installed late 2008 and had experienced something between weekly and daily use over that period with several program installs and uninstalls taking place.

Five sets of experiments were performed. In each case the experiment was conducted with and without Ben-ware running. During the experiments we recorded the proportion of each second that the CPU was active. In each non-idle case a timed script of actions was performed to maximise the consistency between results. The experiments were:

- **System idle:** no activity for ten minutes
- **File copy internal:** 100 files within the laptop
- **File copy USB:** copying 100 files to a USB stick
- **USB insert and removal:** repeated insertion and removal of a USB stick (containing 100 Word documents), twice with correct dismount and eight times just pulling it out
- **Document edit:** Open up document, modify contents, save document, resave document, close document (repeated five times)

Figure 12 shows the average CPU increase per second computed:

$$\frac{\sum_{i=0}^{N} b_i - \sum_{i=0}^{N} n_i}{N}$$

where N is the number of sample points, b_i and n_i are the sample point for Ben-ware running and Ben-ware inactive. All tests added no more than an average of 0.025 CPU seconds – the worst case being the USB device insert, as this required scanning the files on the USB device. Copying to USB increased CPU load by 0.019 seconds as this required accessing the USB device to read directories and files. As a document edit includes probes – which identify the document being opened, written to, and closed – this added an average of 0.012 seconds. An idle state added 0.0096 seconds to the CPU load, which is greater than the file copy internal case as the file copy only ran for a few seconds and wasn't active when the Ben-ware bookkeeping was performed.

Figure 13: CPU load - 10 USB insert / remove

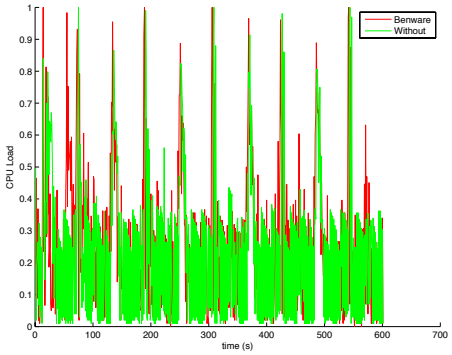

Figure 14: CPU load - Document edit

For the idle and copy cases, the CPU load was increased slightly from the non-Ben-ware case – for the idle case raising the median CPU load from 0.038 to 0.05 seconds, while for Copy to USB this was raised from 0.27 to 0.30 seconds.

Figure 13 shows the CPU load for insertion and removal of an USB memory stick. In this case clear peaks can be seen where the USB stick was inserted with most of the cases for Ben-ware being higher. The system opened up an Explorer window for the newly mounted device, and this accounts for most of the system spike, whilst the spike in the Ben-ware case is increased due to the need to traverse the newly mounted device.

Likewise, Figure 14 depicts the CPU load for editing documents. In this case alternate peaks indicate file opening and file saving. With or without Ben-ware present, this peaks at full CPU load – though only for around one second. Increases from Ben-ware are mostly due to a number of new small peaks of CPU activity, associated with the monitoring of the directory where the document was opened in order to detect file writes.

5.1.2 Memory

Each of the System and User instances of Ben-ware probes are executed as separate Java VM instances. The system instance, running as a system process, consumes 13,268K of memory whilst the User instance, run as the logged in user, requires 10,792K of memory. Thus a total of 24,060 KB (˜23.5 MB) is used by the combination the two instances. Thus on a legacy laptop with only 256MB of RAM, this accounts for 10% of the total memory available. It should be noted that this early prototype of Ben-ware has not been optimised in any way. A full implementation copy would most likely be compiled using a more compact programming language without the need for a Java VM.

5.1.3 Disc Space and Networking

The executables for the Ben-ware probes occupy around 53KB of disc space including the configuration file. The other factors affecting the disc space and network bandwidth used by the probes are determined from the length of each message, the frequency at which each message is generated, and the availability of the network. The first two factors are fairly intuitive, the third requires the knowledge that, if the probes instance can connect to its MLC then it will send all messages over the network and not save them to disc, however if it cannot connect then it will save them to disc for later transmission. Thus each message will always require a little network bandwidth but may only need temporary disc space if the network is not available between message generation and the machine shutting down.

To illustrate the space and networking load, a practise session lasting 249s has 24 heart beat messages, 172 file related messages and 2 removable disc related events. The file containing this log is 19018 bytes long and so this represents an average byte generation rate of 76 bytes/second. Over an eight-hour working day this means a total of 2188800 bytes (2.1MB). If we discount the user activity related messages, leaving only the power, login and heart beat messages then there was 1541 bytes of data generated during the practice session, giving a rate of 6.2 bytes/second. Thus over an 8 hour day this would lead to a total of 178560 bytes (0.17Mb). Thus we can assume that there would be a need for between 0.17MB and 2.1MB of data transfer over the network at a rate of 76 bytes/second, or storage of up to 2.1MB.

The Sony laptop has a 80GB hard drive with 50GB of free space, thus it could comfortable store in excess of 60 years of data at the upper rate.

We must also consider the transmission time and utilisation of the network. Table 2 shows the time to transmit the data, and the network utilisation $\bar{u}$, that an eight-hour day might produce for both the upper and lower data rates over a range of network speeds. The times and utilisations presented in this table assume that 80% of the network bandwidth is available for payload after headers and gaps. If the network bandwidth is in the range of Mbits then we can see that many highly active users could be supported simultaneously. However, it would take fewer than 20 highly active users to completely saturate a 14.4Kbit connection with their Ben-ware network traffic alone.

5.2 MLC

Each of the components within the MLC have been tested for system impact. These tests were carried out on legacy hardware that comprised of a laptop with an Intel Pentium M Processor operating at 1.73Ghz with 512 MB of memory, running the Microsoft Windows XP Professional.

Table 2: Transmission times & network utilisations

Network speed	**Upper Data Rate** (6.2 bytes/s)		**Lower Data Rate** (76 bytes/s)	
	Time	$\bar{u}$	**Time**	$\bar{u}$
14.4Kbit/s	124s	0.43%	1520s	5.3%
28.8Kbit/s	112s	0.22%	760s	2.6%
33.6Kbit/s	53s	0.18%	651s	2.3%
48 Kbit/s	37s	0.13%	456s	1.6%
1Mbit/s	1.7s	0.0059%	21s	0.074%
10Mbit/s	0.17s	0.00059%	0.21s	0.0074%

The performance of different services was observed in an idle state and under load to determine processor and memory usage; figures are an average as observed over a one-hour period. The idle CPU load in all cases is zero, while the CPU load under service conditions is negligible (1-2%) in all cases except for aggregation where it reaches 15% CPU load. As the aggregator will only run briefly once per hour, this is not considered an unreasonable load. In addition, the MLC will normally be run on computers with more computational power thus reducing impact further. Additional metrics are described below.

Table 3: Server's message throughput

	Idle	Load
Processor Usage (%)	0%	2%
Memory Usage (kb)	16944	4300
Message Throughput (msg/sec)	4054	
Max Concurrent Connections	>320	

Table 4: Aggregator's messages generated

	Idle	Load
Processor Usage (%)	0%	15%
Memory Usage (kb)	10224	25000
Messages Generated	2544	
Average Message Size	63	

Table 5: Database's message write rate

	Idle	Load
Processor Usage (%)	0%	1%
Memory Usage (kb)	108	3192
Message Write Rate (msg/sec)	36	

Message throughput (Table 3): This was determined by making a single connection to the server with an instance of the user level probes. This connection was then flooded with 9,244 messages of synthetic data developed for this experiment, and the time taken to send these messages was measured. Maximum concurrent connections were determined with the use of a telnet client. The limit regarding the number of connections is due to the available client computer resources; therefore the true maximum concurrent connections for the server is actually higher.

Messages generated (Table 4): This represents the number of aggregate messages generated from one year of synthetic data for a single user, for further transmission to the AI component. In our experiment, these data points were computed in 2 minutes and 4 seconds. Average message size is the average size of the messages that would be forwarded to the AI. It should be noted that the processing of data would normally happen in an (hourly) real-time manner. Thus the processing of the synthetic log into aggregates would take on average no more than 0.067 seconds per aggregation (assuming a user was only present for five days per week, 46 weeks per year, and eight hours per day).

Message write rate (Table 5): This is the rate at which MySQL wrote message data to the database after the server process had accepted data and performed an `insert` command. Although the message-writing rate here is much lower than the possible data generation rate, in practice this would not be an issue as the MLC would need to receive over 36 messages per second before it would overload the MySQL server. If this were the case, the MLC could delegate some of its users to other computers.

Table 6: User: Jo – VERY HIGH RISK threshold

Test Dataset	α	β	Ω
Jo_Bad1	1	1	0
Jo_Bad2	85	82	1
Jo_Bad2_gaps	76	72	0
Jo_Bad3	69	66	1

Table 7: User: Jo – HIGH RISK threshold

Test Dataset	α	β	Ω
Jo_Bad1	1	1	7
Jo_Bad2	85	82	2
Jo_Bad2_gaps	76	72	2
Jo_Bad3	69	66	2

5.3 Artificial Intelligence

All algorithms have been implemented using Java. They have been tested on both Linux and Windows 7 and 8 platforms. Before deploying in a new organisation, the algorithms have to be re-trained, as it needs to construct data models for the workforce. During this process, a model reflecting the normal behaviour of each employee will be learnt from the previously recorded usage patterns. The time-span of the training sample is dependent on the diversity of user behaviour. We believe that for an employee whose daily behaviour is fairly consistent, around 3 months of data (approximately 60 data points) is sufficient.

However, for an employee who shows high degree of variability in computer usage behaviour, it is prudent to train the system with more data (approximately 200-250 data points). In situations where historical data is not available for an employee, it is possible to train the system using data from another employee who performs similar job functions. The system could be re-trained using up-to-date information regularly (optimally once every 6 months). Since the training module contains in-built cross-validation functions that divide the training set into different combinations of training and test data, selection of optimum parameters for SVDD or number of clusters does not require inputs from the systems administrator.

Once the system is trained, it can be used to identify suspicious behaviour of any of the employees, as it will hold the models of all users in memory. A decision on each user will be taken once a day under the normal circumstances and most probably on his/her main PC. As a result, no single computer will be burdened with anomaly detection. There will only be one instance of the software running on a computer, as it does not require multiple instances to handle different users.

Tables 6–8 show the results obtained using test data for a single user under different test scenarios. Here α is the number of true positives, β the number of true positives identified and Ω the number of false positives. As the final result is presented as NO RISK, MODERATE RISK, HIGH RISK or VERY HIGH RISK, these tables indicate when this threshold has been surpassed. It should be noted that for each 'Bad' action type there is only one invocation of this bad activity. Therefore the difference between α and β represents the number of days between a bad action commencing and the AI identifying the user change.

Table 6 illustrates the situations where user Jo exhibits the simulated bad behaviour. The number of true positives (α) indicates the number of bad events within the data set, for example Jo_Bad1 indicates that on only one day did Jo

Table 8: User: Jo – MODERATE RISK threshold

Test Dataset	α	β	Ω
Jo_Bad1	1	1	24
Jo_Bad2	85	82	10
Jo_Bad2_gaps	76	72	10
Jo_Bad3	69	66	14

steal a large number of files. The number of true positives identified (β) indicates the number of 'bad' days that the AI picked up. The number of false positives (Ω) indicates the number of days where the AI incorrectly identified a theft of files. In all cases the total number of samples is the number of days that were contained within the user trace – this is 391 as we assume here that staff do not work at the weekend. For Jo_Bad2 and Jo_Bad3 it can be seen that three days after Jo started stealing files, the AI is able to detect this. However, in both of these cases, the number of files stolen is two per day (with Bad3 increasing by one per day), thus the system is sensitive to small variations in usage patterns. In both of these cases, the number of false positives is just one – although each of these cases would need human investigation, this number is relatively small. For the case of constant theft with gaps, the system requires an extra day (i.e. four days) to realise that files are being stolen. However, it does not present false positives.

Traces for each user and each bad pattern were generated and processed by the AI system. Figure 15 shows the average time to identify the different bad actions from the set of users. In each case a standard deviation is also marked on the bar. In all cases of Bad1 (single theft) the act was picked up on the day that the action was performed. In all cases the number of files stolen was 100, however, some of the users would normally take files home with them. Likewise for the Bad2 case (constant theft) the AI was able to detect this in exactly three days for all users. When we allow gaps to occur in the theft of these files (Bad2 Gaps) the average number of days comes down slightly though now there is some variation in the number of days. This variation comes from user Sally for whom the system takes only two days to identify the Bad2 Gaps case. In all of these cases the number of files stolen per day is just two. For the case where the number of files stolen per day increases, the average is the highest ($4\frac{1}{3}$). This variation is due to Sally and Bob who do not normally take files home.

Figure 16 shows the average number of false positives – cases where the AI has incorrectly identified an anomalous activity where none exists – along with the standard deviations. For both Bad1 and Bad3, no false positives were present. However, for both cases of Bad2, false positives were present. These were due to users Jo and Ben. For the case of Ben (where he is willing to break the rules to get the job done), he is more likely to perform activities that could be seen as malicious – Ben is a system technician who is more likely to perform activities which are abnormal.

In Tables 7 and 8, we investigate the effect of reducing the threshold to trigger from VERY-HIGH RISK down to HIGH RISK and MODERATE RISK respectively. In both cases, the time to identify abnormal behavior remains the same. However, the number of false positives increases – slightly for the HIGH RISK case and markedly for the MODERATE RISK case. Although these reductions of threshold do not increase the number of cases where abnormal behavior is detected (as all cases were detected before), it does provide

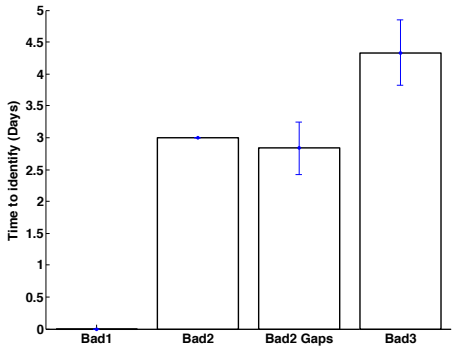

Figure 15: Average time to identify bad action

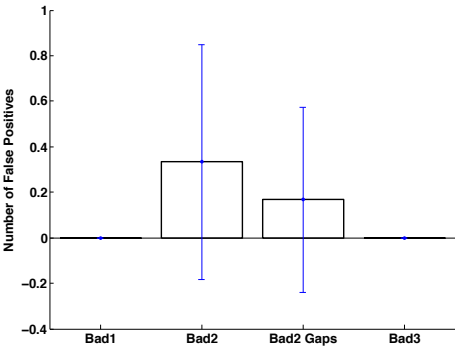

Figure 16: Average occurrence of false positives

the opportunity to see that the user is acting out of profile. However, as this leads to higher levels of false positives, we should not treat these numbers too seriously. Nonetheless, a user spending significant time in these lower categories could indicate a move towards malicious actions.

6. DISCUSSION

The Ben-ware system combines artificial intelligence with machine learning and various (real) 'human factors' information – that the organisation may already know about the individual – to identify anomalous behaviours that threaten the work of closed organisations. In addition to the technological 'proof of concept' issue – which has been the main focus of this paper – there is also the additional problem of the ethical conflicts that such a system introduces into the frame. Part of the answer to the question of ethics is given by the closed nature of the organisations and their specific information security policies and employment contracts. Moreover, the two should be linked in order to align the expectations of both the employees and the organisation. They should also balance the strict needs of security with a flexibility to provide employees the freedom to work creatively in order to achieve the organisation's goals.

We believe that Ben-ware is most effective where the two overlap. Where Ben-ware will not work is in an organisation where the information system is open, however, we believe that the next generation of Ben-ware will be able to be applied to parts of such a system – especially that part which contains confidential or restricted information and which has to be protected from being leaked. Again, where this is the case, information policies and employment contracts will have to be co-ordinated to ensure that the expectations of both parties are similar. Such discussions inevitably frame the proof of concept work, which this article has focused upon.

7. CONCLUSION AND FUTURE WORK

We have developed a successful prototype for evaluating insider attack threats against an organisation where the attacker intends to 'steal' files and data. The system is capable of identifying – within three to four days (or immediately in the case of a large file theft) – when a particular user is behaving in an anomalous manner, thus leading to a suspicion of becoming an insider threat.

We have achieved a good detection rates – identifying all occurrences of bad activity – along with low false-positives – an average of less than 0.4 false-positives. Furthermore, the overheads incurred by the system are not only distributed over the computers set, but also are low in each case. The overheads account for a small load increase (less than 2.5% CPU load, 1-2% for the MLC) apart from the aggregator, which requires 15% CPU load whilst functioning. However, as the aggregator will only run once per hour for a fraction of a second, the impact is not as significant. The memory footprint for the code is also negligible (less than 10% on a 256MB system). This is more impressive when it is noted that the hardware and devices used for this analysis are legacy equipment, most being over eight years old, and that the code has not been optimised in any way. We therefore see Ben-ware as a scalable and promising system capable of detecting insider threats.

To improve Ben-ware's accuracy, we plan to give each user-activity a risk value, and each user will be given a daily (or monthly) allowance on how much risk value he/she can use. For example, the insertion of a USB storage device and the reading of a sensitive file will have their own risk values but the combination of the two may yield a higher risk value. This approach allows us to employ multiple means of capturing anomalous activities, which will lead to a more robust and reliable means for detecting insider threats.

At the moment, only a small dataset has been used to illustrate the feasibility of the Ben-ware approach. The distributed nature of the Ben-ware system will allow the system to scale linearly, meaning the load on available resources should be unchanged (or only by a small amount) even when there are hundreds or thousands of users being monitored. It is true that there is a possibility of a bias in the synthetic data used in the present experiment, hence future work will be carried out to deploy Ben-ware in a real environment, with real human factors (e.g. data derived from user's personality test as well as HR and disciplinary records) being included in the computation of the user profiles.

8. ACKNOWLEDGMENTS

We would like to thank UK Defence Science and Technology Laboratory (DSTLX1000088892) for funding this work.

9. REFERENCES

[1] B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, and A. Sheth. An ontological approach to the document access problem of insider threat. In P. Kantor, G. Muresan, F. Roberts, D. Zeng, F.-Y. Wang, H. Chen, and R. Merkle, editors, *Intelligence and Security Informatics*, volume 3495 of *LNCS*, pages 486–491. 2005.

[2] M. Bishop and C. Gates. Defining the insider threat. In *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, CSIIRW '08, pages 15:1–15:3, New York, NY, USA, 2008.

[3] B. Bowen, M. Ben Salem, S. Hershkop, A. Keromytis, and S. Stolfo. Designing host and network sensors to mitigate the insider threat. *Security Privacy, IEEE*, 7(6):22–29, Nov 2009.

[4] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut. Proactive insider threat detection through graph learning and psychological context. In *IEEE Symposium on Security and Privacy Workshops'12*, pages 142–149, 2012.

[5] D. Caputo, M. Maloof, and G. Stephens. Detecting insider theft of trade secrets. *Security Privacy, IEEE*, 7(6):14–21, Nov 2009.

[6] F. L. Greitzer and R. E. Hohimer. Modeling Human Behavior to Anticipate Insider Attacks. *Journal of Strategic Security*, 4(2):25–48, June 2011.

[7] M. Kandias, A. Mylonas, N. Virvilis, M. Theoharidou, and D. Gritzalis. An insider threat prediction model. In S. Katsikas, J. Lopez, and M. Soriano, editors, *Trust, Privacy and Security in Digital Business*, volume 6264 of *Lecture Notes in Computer Science*, pages 26–37. Springer Berlin Heidelberg, 2010.

[8] K. D. Loch, H. H. Carr, and M. E. Warkentin. Threats to information systems: Today's reality, yesterday's understanding. *MIS Quarterly*, 16(2):173–186, 1992.

[9] G. Magklaras and S. Furnell. A preliminary model of end user sophistication for insider threat prediction in IT systems. *Computers&Security*, 24(5):371–380, 2005.

[10] J. Murphy, V. Berk, and I. Gregorio-de Souza. Decision support procedure in the insider threat domain. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, pages 159–163, May 2012.

[11] J. Nurse, P. Legg, O. Buckley, I. Agrafiotis, G. Wright, M. Whitty, D. Upton, M. Goldsmith, and S. Creese. A critical reflection on the threat from human insiders – its nature, industry perceptions, and detection approaches. In T. Tryfonas and I. Askoxylakis, editors, *Human Aspects of Information Security, Privacy, and Trust*, volume 8533 of *LNCS*, pages 270–281. 2014.

[12] E. Pauwels and O. Ambekar. One class classification for anomaly detection: Support vector data description revisited. In P. Perner, editor, *Advances in Data Mining. Applications and Theoretical Aspects*, volume 6870 of *Lecture Notes in Computer Science*, pages 25–39. Springer Berlin Heidelberg, 2011.

[13] L. Spitzner. Honeypots: catching the insider threat. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 170–179, Dec 2003.

[14] D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[15] P. Thompson. Weak models for insider threat detection. In *Proceedings of Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, volume 5403, pages 40–48, 2004.

[16] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, USA, 1995.

[17] D. Wall. Enemies within: redefining the insider threat in organizational security policy. *Security journal.*, 26(2):107–124, April 2013.